Kabira

always on

Kabira Transaction Platform

Anatomy of High Performance

KABIRA WHITEPAPER 01.2007

Table of Contents

The Context for Kabira

The Kabira Transaction Platform™ delivers high-performance and scalable transaction processing at a low cost per transaction, using model-driven development techniques to accelerate time to market and manage complexity.

Kabira starts with a high-speed, standards-based software infrastructure. The underlying technology can support tens-of-thousands of events per second on commodity UNIX platforms. The Kabira Transaction Platform supports high-volume real-time events, transactions, connections to other systems, active caching and handling of bursts of traffic, as well as other services. This infrastructure was designed to connect to and across all layers in a customer network. It can deal with slower speed, complex data and applications at the Business Systems layer. It can deal with higher speed, less complex data at the Operations Systems layer, such as management, alert and control applications. It can connect all the way down to the hardware or network element layer in both directions - issuing commands directly to hardware and accepting events, alerts and status from hardware or network level, real-time software.

Perhaps the most fundamental part of understanding the Kabira Transaction Platform is appreciating the meaning of the word "transaction". To sales staff, transactions signify a stream of orders. To a CFO, transactions may mean credit card transactions or accounting journal entries. To software professionals, the word frequently brings to mind database transactions of the sort that keep information in a consistent form.

The concept of "transaction" that is relevant to the Kabira Transaction Platform is general enough to encompass transactions in the broadest sense:

- Kabira transactions include financial transactions, credit card payments, cell phone text messages, accounting journal entries, database transactions, and many other ways of accepting information, usually through a stream of messages.
- Transaction processing within Kabira refers to the very fast processing and recording of huge volumes of information as it comes in and then sending out messages to elicit the proper response.
- Kabira transactions have all of the 'ACID' properties that are expected from software professionals

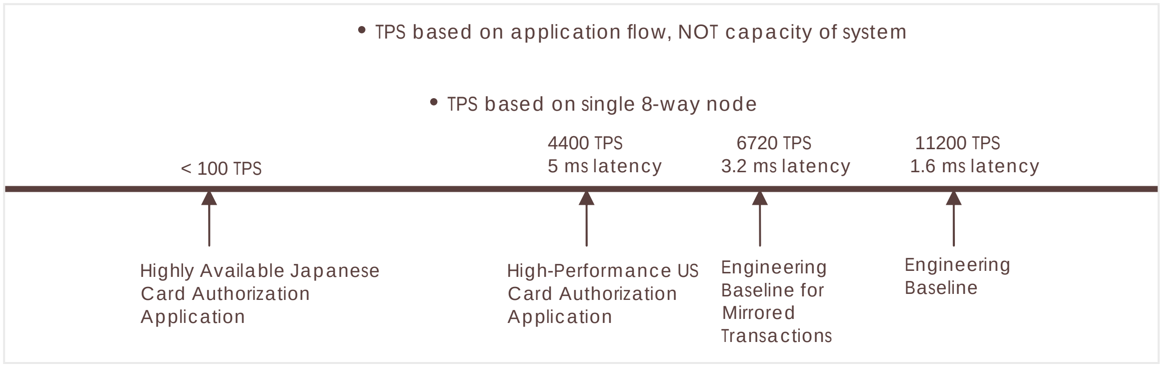
• TPS based on application flow, NOT capacity of system
• TPS based on single 8-way node
< 100 TPS
4400 TPS
5 ms latency
6720 TPS
3.2 ms latency
11200 TPS
1.6 ms latency
Highly Available Japanese Card Authorization Application
High-Performance US Card Authorization Application
Engineering Baseline for Mirrored Transactions
Engineering Baseline

Figure 1: Kabira Performance Benchmarks

To achieve the highest performance, the Kabira Transaction Platform uses a memory-resident database to execute the fastest possible reads and updates. Kabira's in-memory database can act as either a system of record or be replicated or distributed to other repositories such as databases or files. Kabira can achieve throughput of tens of thousands of transactions per second. The Kabira Transaction Platform can process messages with the reliability normally associated with super-expensive, specialized, high-availability hardware.

Kabira's development methods hide implementation complexity. Using model-driven techniques, programmers can develop and revise highly complex applications in a fraction of the time required by traditional implementation methods—bringing new products and services rapidly to market. Applications can be configured and updated without downtime. And perhaps best of all, Kabira software is designed to deliver all this power—and extreme scalability—on low-cost commodity computers.

This white paper reviews the trends that are driving the need for increasingly high-volume transaction processing and then describes the Kabira Transaction Platform in detail, including the solution architecture, scalability and high-availability techniques, development and testing frameworks, configuration, deployment strategies, and run-time monitoring.

Living in a Real-Time World

High-performance transaction processing used to be a rare phenomenon, utilized only in extreme environments by the largest companies. But in recent years, the Internet has opened the door to the arrival of global customers in quantity through e-commerce sites, call centers, and other forms of direct interaction. Business-to-business relationships are intermediated by direct computer-to-computer interaction, frequently based on Web services. Content delivery and mediation for services must take place in real-time. This bulge in transaction traffic follows the same pattern that has transformed the telecommunications industry from a few providers of old-style, fixed local and long-distance calling services into a competitive field of real-time enterprises offering wireless mobile plans for delivery of complex, combined data, voice and video content.

The requirements of global and real-time transaction processing are becoming the norm, driving enterprises to seek out IT systems whose architectures can handle skyrocketing transaction volumes at the lowest possible cost per transaction, in a manner that allows for flexibility and agility in service offerings. Flexibility, high performance and low cost constitute a new transaction-processing triangle that confounds solutions and architectures designed on proprietary systems as recently as a decade ago.

Previous Transaction Processing Pattern and Business Model, Telecom Industry:

- One call data record (CDR) written when a call started, and one when a call ended
- CDR records were used at the end of the month to create a bill for the customer
- Batch processing was perfectly adequate
- Revenue-per-call could be relatively high - several dollars or more
- Business model and service offering remained static for years

Today's Transaction Processing Pattern and Business Model, Telecom Industry:

- Multiple vendors or carriers are involved in each cell phone call; each of them tracks activity and grants permissions.
- Transaction volume has exploded. Each call may generate as many as 100 CDRs to track various transactions.
- The number of services has grown from a simple voice call to text messaging, Internet access, video, real-time data and special purpose e-commerce functions
- Approval to use a service must be granted in real time
- Revenue-per-transaction is measured in cents or micro-cents
- Business model and service offerings change quarterly or more frequently

Converging Platforms and Architectures

In the business world, the elements of the economic triangle (fast-to-market, high-quality, low-cost) correlate to the elements of the transaction processing triangle: flexibility, high-volume/high performance and low cost-per-transaction/low TCO.

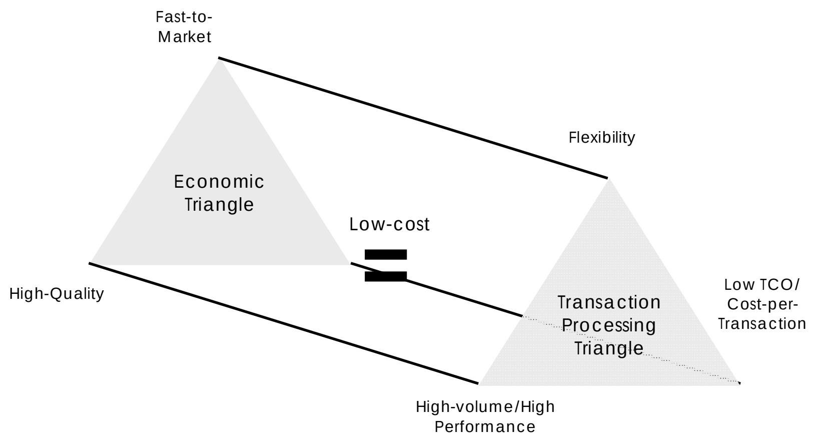
Fast-to-Market
Economic Triangle
Flexibility
Low-cost
=
High-Quality
Transaction Processing Triangle
Low TCO/ Cost-per-Transaction
High-volume/High Performance

Figure 2: Economic Triangle/Transaction Processing Triangle

In an economic triangle, one is typically allowed to pick only two of the three sides of the triangle and must compromise on the third. You may choose a fast-to-market and low-cost solution, but it won't be high quality. A high quality and fast-to-market solution won't be cheap. Only the most innovative and successful products in their domain find a way to deliver on all three sides of the economic triangle.

Current transaction platform solutions fall short of delivering on all three legs of transaction processing triangle:

- High-availability hardware systems such as HP NonStop, IBM's CICS and Base24 are high volume, but are not low-cost or flexible.
- Specialized transaction processing systems like BEA Tuxedo are medium-volume, cheaper than high-availability systems, but are not flexible.
- Application servers such as J2EE/JTS may be less expensive than high-availability hardware systems and specialized transaction processing systems, and they are flexible. However, they are not high-performance, nor are they highly available.

In attempts to meet the demands of the transaction processing triangle, some vendors are merging specialized transaction processing systems with application servers or installing application servers on highly available hardware, and so forth. Almost none of these attempts have been successful.

Other vendors seek to solve the problem on an architectural level, by combining the flexibility offered by service-oriented architecture (SOA) with the increased automation and decoupled nature of event-driven architecture (EDA). SOA-based solutions are delivered as sets of services that can be combined and recombined to meet new requirements. Using EDA, solutions are increasingly automated, bringing in human intervention only when an exception arises that is beyond the scope of an automated solution.

The Kabira Answer

No vendor except Kabira has produced a transaction platform that actually delivers on all three aspects of the transaction processing triangle. To achieve this, Kabira combines:

- The data access speed of an in-memory database
- The flexible logic of an application server
- The high-speed transport and dynamic routing of EAI middleware
- The transaction processing speed of specialized systems
- The resilience and fault tolerance of high-availability hardware

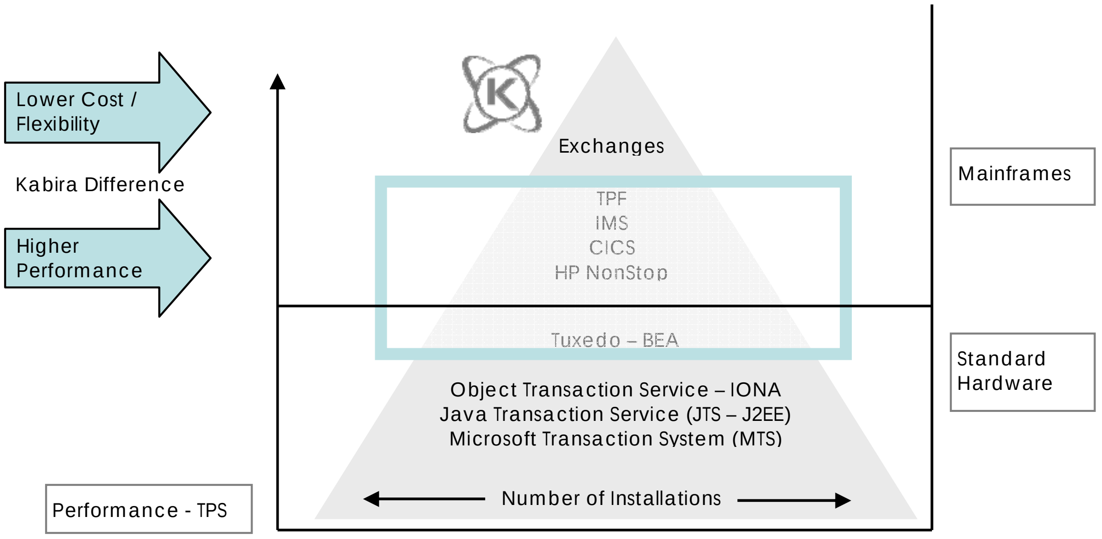

Figure 3: Kabira Competitive Positioning

Model-driven development is the only way to conquer the complexity of high-performance computing and allow all of these aspects to be combined into one system. Building a solution on the Kabira Transaction Platform lets programmers create a model in an Eclipse environment, using an Action Language to specify detailed logic. The model specifies what the system will do by describing objects and their relationships. Once this is done, the Kabira Transaction Platform compiles that model into a solution and optimizes it at every level, combining various subsystems for data management, process management, integration and configuration, and providing other key transactional services needed for transaction processing. This sort of optimization is simply not possible with traditional API and programming language implementation techniques.

The rest of this paper explains how the Kabira Transaction Platform is engineered to fulfill all constituents of the transaction processing triangle, enabling businesses to offer a stream of evolving high-quality services in the marketplace, at a fraction of the cost of the alternatives.

The Kabira Transaction Platform

The architecture of the Kabira Transaction Platform establishes a fundamental set of capabilities and services that can be reused and combined in different ways to create flexible, high-performance solutions. The work of every Kabira-based solution is performed by the following elements:

- Kabira Infrastructure System™ (KIS): the part of the platform focused on data management
- Kabira Transaction Switch™ (KTS): the part of the platform focused on process management
- Kabira Channels: a high-speed adapter framework for bringing data into the platform from external sources and sending it back out again, usually through a stream of messages
- Kabira Configuration Management: the part of the platform that provides an abstraction of the system landscape
- Kabira Operations Management: the part of the platform that monitors real-time operations and allows a Kabira solution to be dynamically reconfigured
- Kabira Transactional Services: specialized elements such as deadlock detection, mirroring, caching, pooling, transactional replication, memory management and routing functionality used to create high-performance transaction processing systems.

The application model dictates how each of these platform elements is combined and optimized. Figure 4 provides a general idea of how these parts fit together into a Kabira engine, which is the fundamental unit of deployment.

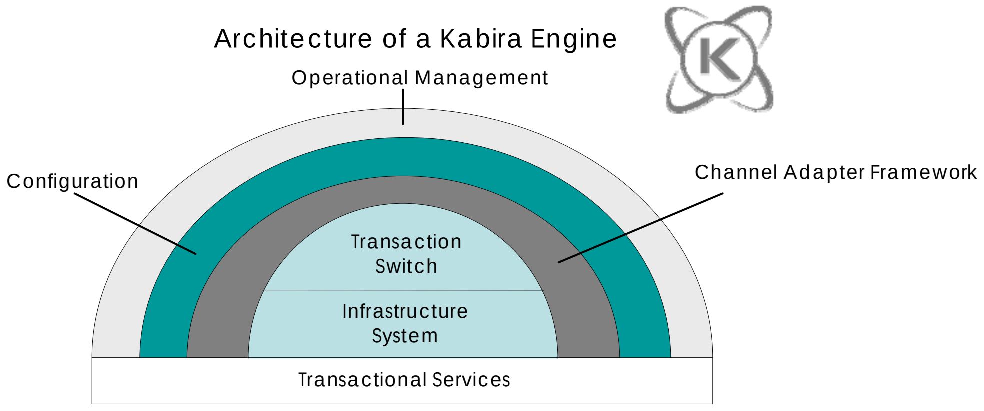

Figure 4: Kabira Engine Architecture

The Configuration and Operational Management layers do similar jobs as the same layers in other software, providing ways to allow Kabira to run in many different situations just by changing settings. The details of both of these layers will be discussed later.

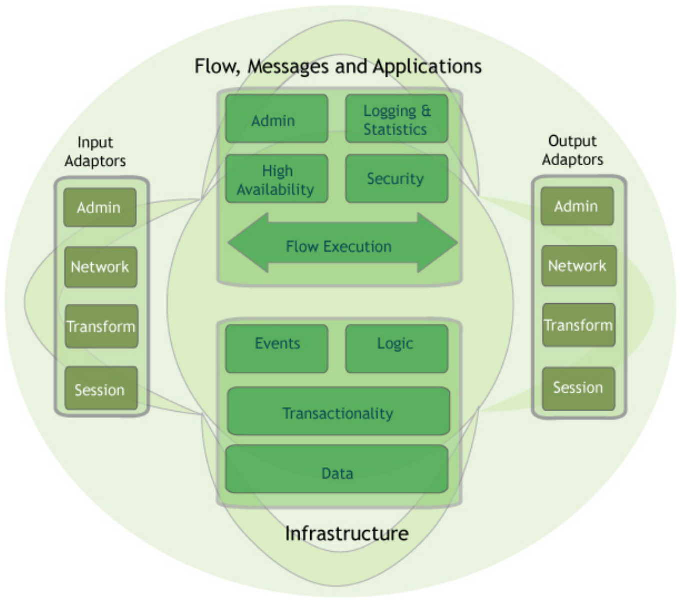

Figure 5: Kabira Flows

The high-speed Channel Adapter framework moves data in and out of Kabira. One of the foundations of Kabira's high-speed transaction processing is that all data in the KTS and KIS layers is stored in shared memory, and is normalized to a standard format. All of the logic and processing that converts data to and from this standard format exists in the channel adapters. Channels ensure that you will not have to change the application to function with a new (or newer version) of a communication protocol –you will only need a new channel adapter. Channel adapters are modeled rather than coded, using traditional programming languages. This allows the compiler to analyze and optimize the data transformations that take place in the adapters.

The KTS layer is real-time application server focused on process management. When a Kabira application is modeled in Eclipse, a series of objects is created and their behavior and the interactions between them are described in the model. To automate a business process, information flows from one object to another and the state of the process must be maintained. To fully take advantage of modern hardware that allows for the execution of many simultaneous threads, the process flows in a model must be analyzed to determine which parts of the process are parallel and which are serial. The KTS layer performs this analysis and then provides all the plumbing to synchronize parallel execution when possible.

The KIS layer can be thought of as a real-time application server focused on data management. When thousands of transactions per second are streaming through the engines in a Kabira node, the shared memory acts as a database. Given that each engine is running multiple threads and each engine is a separate, multi-threaded process, keeping the data consistent requires strict control of access to the data. But at the same time, all processing must proceed as quickly as possible. The KIS layer provides the locking, event procession, data replication, and synchronization mechanisms used to turn shared memory into a high-speed repository for transaction processing.

Transactional services include utility functionality for everything that is needed in a high-speed transaction processing platform, from real-time loading of components in engines to message queuing to error reporting.

Packaging Solutions for Kabira Deployment

Solutions built on the Kabira Transaction Platform feature the following structure:

- Components (such as objects, adapters, utilities, and so forth) are modeled in UML and combined to create an engine.
- Engines communicate to the outside world using channel adapters that send and receive messages, process data using components, and store and retrieve data from shared memory. Each engine is multi-threaded and runs in its own UNIX process.
- Configuration information is used by an engine to determine sources and destinations for messages.

- Nodes are a collection of engines working together on one area of shared memory, which plays the role of a database (as shown in Figure 6). Any number of engines, each playing similar or different roles, can be part of a node.

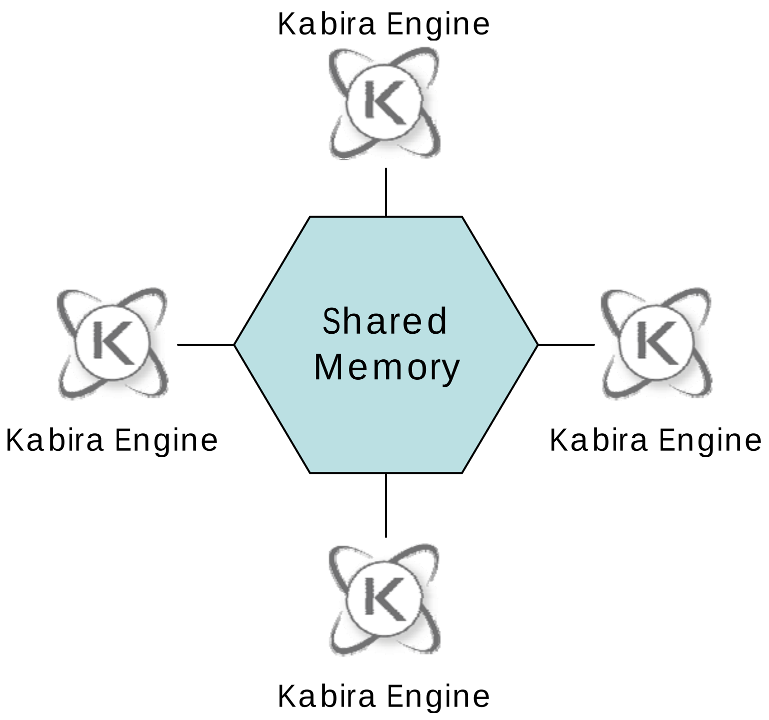

Figure 6: Kabira Node Architecture

Integration Mechanisms: Kabira Channel Framework

The Kabira Channel Adapter framework is the integration workhorse of the Kabira Transaction Platform. Like most transaction platforms, information passes to and from Kabira in the form of messages. Even when external APIs are used to gather information from databases or special purpose external systems, the information gathered can be considered a stream of messages. Kabira channels are adapters built using the framework. Channels move data to and from databases and external systems of record, Web services provided by other applications, and message queues from EAI systems at very high speeds.

The Channel Adapter framework transforms external messages into the normalized information format used by the Kabira Transaction Platform. Kabira uses modeling for everything, including the construction of its channel adapters, unlike other adapter frameworks that are based on coding in languages like Java or C++. This allows for optimization of message movement to and from Kabira, taking advantage of parallel processing, queue mechanisms, and other aspects of the Kabira Transaction Platform without conscious intervention by the system developers. Modeling channel adapters also increases developer productivity and decreases maintenance.

Kabira optionally supports transactional writing of some data either to external databases or to files in the file system.

Scaling Mechanisms Achieve Seamless Scaling at Lowest Cost

The extreme transaction processing that characterizes digital enterprise today demands the ability to scale – perhaps exponentially—as transaction complexity and numbers of users grow. Kabira's Transaction Platform permits massive and seamless scaling as business needs increase and supports both single and distributed scaling mechanisms. This provides application designers with the flexibility to make trade-offs between cost, manageability, and redundancy as required.

Kabira achieves scalability at the lowest possible cost in three decisive ways. The first is Kabira's use of low-cost commodity hardware. Models used to define Kabira's solutions can be compiled to run on popular flavors of UNIX. The Kabira Transaction Platform is designed to scale on both a single platform and across multiple platforms. Thus, you may choose to deploy Kabira on the system that fits your processing needs: applications hosted on the Kabira Transaction Platform can be deployed on large multiprocessor machines or on many smaller machines to meet application performance requirements. Less processor-intensive or memory-intensive Kabira nodes can be deployed on low-cost hardware running Linux, while Kabira nodes requiring the most powerful processors and the largest memory spaces can run naturally on the most powerful hardware.

Platform Scaling

Massively scalable to handle rapid growth in transaction volume, the second way that Kabira can scale is to scale "up" linearly with CPUs and clock speed. The Kabira Transaction Platform is designed to take advantage of the CPU, memory and disk resources available on its host platform. As CPUs, threads, real memory and disk space increase, the Kabira runtime will scale transparently to the application.

Kabira achieves CPU scaling through the use of operating system-level threading. The total number of threads used by Kabira is optimized to minimize "empty CPU cycles" caused by excessive thread context switching. This ensures that CPUs are kept busy performing application work, not on operating system housekeeping. The architecture does not just "throw threads at the problem" to simplify the implementation.
In addition, the Kabira Transaction Platform does not perform global locking. All locking of shared resources is designed to minimize lock contention. This is accomplished by eliminating global resources that must be locked by all threads before any work can be performed.

This vertical scalability power derives from the Kabira platform's unique consolidation of data, event and logic processing into the 64-bit shared memory, and its kernel-threaded event execution architecture, which gives the Transaction Platform the power to take maximum advantage of the computers on which it runs. A Kabira solution can utilize as much memory as will ever be available on a computer. Placing all of the data in memory enables incredibly fast processing. Kabira's unique linear vertical scalability allows applications to scale from Proof-of-Concept to full-scale production without performance re-engineering.

Distributed System Scaling

Application scaling can also occur in a third way as Kabira scales “out” horizontally with the introduction of additional servers. Kabira’s node architecture allows as many nodes as needed to run on as many computers as desired. The message traffic is spread across the nodes by a special engine called a distribution engine (as shown in Figure 7).

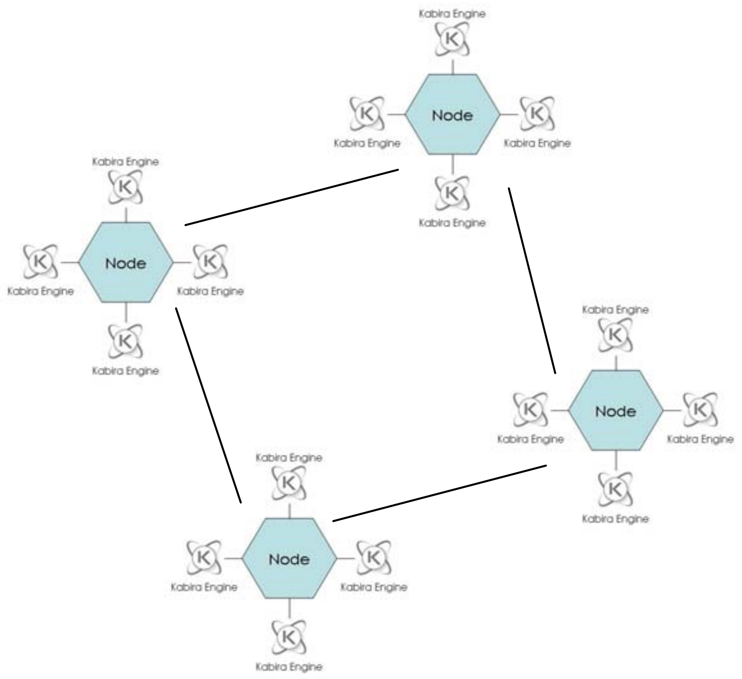

Figure 7: Kabira Scalability and HA Architecture

A distribution engine allows messages to be spread across other nodes that are running on the same computer or different computers in many different ways. For example, for a given stock-trading application, all of the ticker symbols from A to K could be on one node and the tickers from L to Z could be on another. The Kabira Transaction Platform’s ability to perform dynamic configuration via the High Availability Component improves this picture even further. If a particular stock is trading heavily, it is possible to route traffic just for that ticker to another node to better balance the load. This sort of rerouting can take place on the fly with no downtime.

Contrast this with the use of middleware and database servers to deploy distributed applications; these require the applications themselves to incorporate distribution functionality. In other words, middleware and database server support for distributed applications is little more than simply transporting data from one machine to another, which is the simplest part of building distributed applications.

High-Availability Mechanisms and Methods

The HA Component gives an enterprise business system “five 9s” (99.999%) of availability without reliance upon redundant clusterware, transaction monitors or databases. Five 9s is a mainframe-class, high-speed, high traffic system that has no more than five minutes of downtime per year. Kabira High Availability provides a low-latency system, where failovers are transparent to users, there is no interruption of work, no transactions lost and backup and recovery functions occur with no degradation in performance. The Kabira Transaction Platform uses many of the battle-tested techniques for achieving high availability.

However, unlike almost all other solutions, Kabira implements high availability completely in software without using specialized hardware, shared or clustered disks, or redundant hardware lying idle in stand-by mode. Transaction routing ensures that business applications are always available, even if there is a hardware, operating system or application failure.

The first requirement for highly available systems is that all data must be stored in different places. Traditional highly available systems are like RAID 5 storage devices that spread data out over several disks, making sure that all data is available on at least two physical drives at all times. The Kabira Transaction Platform utilizes transactional replication and mirroring functions to accomplish this.

The platform guarantees data integrity by routing each transaction to an active copy of the data. Key stateful objects can be tagged to be highly available so that in the event of a transaction failure, the Kabira High Availability Component’s Message Router forces a switchover to a designated backup server, which then becomes active and assumes responsibility for completing all transactions until the primary server is restored. The system continues to process every transaction in real-time, and failover is transparent to users. Object partitions in the HA Component assure data integrity by having a single master copy of each instance and by avoiding distributed locks.

The transactional part of the replication makes sure that the write on the primary system is not considered complete until the replication is complete. Any node can play the role of a primary node, a backup node, or both at the same time.

The Kabira Transaction Platform also features a variety of high-availability mechanisms such as the ability to queue bursts of transactions during high-volume periods and the ability to re-process transactions that failed in the middle of a transaction.

Kabira HA Protects Both Data and Applications

When the Kabira High Availability Component is enabled, HA and recovery logic are inherent in every application built on the Kabira Transaction Platform, providing transparent application recovery support and saving costly IT resources and time, since programmers need not write HA-aware code into each application.

Kabira’s memory-resident transactions and processing ensure memory-speed recovery from failure. Moreover, high-speed rollback and recovery occurs not only for all data associated with an application, but also the current processing state at time of failure; thus, applications restart from the last successful processing step.

Security Component

Kabira's Transaction Platform features a powerful and flexible Security Component that ensures the confidentiality, integrity and security of all data and mission-critical applications deployed on the Kabira platform. Security-enabled Kabira systems perform with very high speed and efficiency because users enable security only for those operations they wish to secure—there is no performance impact to operations users don't wish to secure. Endowing any application with Kabira Security is as easy as turning on the Security Component – no coding required.

The Kabira Security Component provides users with fine-grained control over rules of access, allows them to add security functionality retroactively to previously unsecured applications, provides support for existing security technologies via the Kabira Security Services Layer (KSSL) and offers the highest degree of security with the lowest performance impact.

For more detailed information about the Kabira Transaction Platform's security features and functions, refer to Kabira's Security Component brochure at www.kabira.com

Flexibility Mechanisms and Methods

The Kabira Transaction Platform was designed to be flexible enough to handle the real-world challenges of high-volume transaction processing. Kabira offers platform independence, supporting Sun/Solaris, HP/Linux, Hitachi/Linux, and AMD Opteron/Linux.

New Kabira components can be loaded into an engine while the rest of the engine continues running. The new component can go live with virtually no impact on an operating system, allowing bug fixes or new versions to come into production without scheduled downtime.

Kabira's configuration mechanisms are similarly flexible. When a Kabira engine is loaded, the engine looks for configuration information that describes the location of other Kabira nodes to communicate with, and for any external services that will be used. Configuration information can be changed on the fly. Failsafe mechanisms exist to prevent shutting down a connection to a node while the other node is still active.

Configuring for High Availability (such as determining which node is going to handle which messages), is also configurable at runtime.

Non-Disruptive Online Change Management

Uncontrolled system changes are a frequent cause of system outages. A company must be able to build, test and deploy new versions of mission-critical applications as often as necessary—while relying on continuous systems and transaction operations. Kabira change management functions ensure system stability during major system changes, including during system configuration changes and when adding or deleting hardware devices from the running system. A number of reconfiguration tasks can also be controlled at the application level in order to maintain, repair or upgrade elements of a system without needing to shut down or restart.

Kabira Solution Architecture

From a technical perspective, components, engines, and nodes tell the story of how Kabira works, but applications and solutions built with the Kabira Transaction Platform can organize functionality in several different ways.

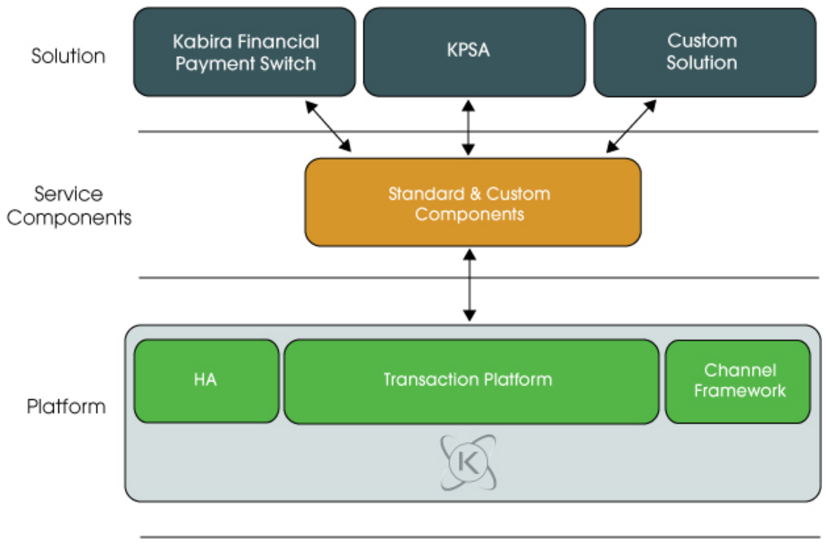

Figure 8: Kabira Architecture Showing Kqbira Solution Architecture

Kabira Service Components are special purpose collections of Kabira components, engines, and nodes that are dedicated to executing a particular function – specific needs for your line of business. For example, the Kabira Payment Component contains functionality that helps process electronic payments. The components were developed as Kabira repeatedly solved problems in certain domain areas and found ways to speed development through reuse. Both Kabira development staff and Kabira customers make use of service components.

Kabira Solutions are enterprise software products built using the Kabira Transaction Platform. Kabira solutions exist for payment processing, for telecommunications provisioning, and several other business areas. While Kabira Solutions are built using the Eclipse-based, model-driven technique, they were created to be configurable solutions like any other enterprise software. Modeling in Eclipse can be used to extend the functionality of a Kabira Solution.

Integration with Systems of Record

One of the most important aspects of Kabira Solutions is the different ways they interact with data from systems of record containing customer or account information. At first, some people think of the Kabira Transaction Platform primarily as a high-performance cache, because of the way the Platform uses messaging and a memory-resident database. However, through the use of the Channel Adapter Framework and Kabira's mechanism for transactionally storing data in various repositories, Kabira Solutions can operate as systems of record and manage important information.

Kabira Solutions and the Kabira Transaction Platform interact with data in other systems in the following patterns:

- As a system of record: In this pattern, Kabira becomes the master system of record and may use a persistent storage mechanism, such as a database, as a permanent repository. Any other systems that need the data stored in Kabira can ask for it through message-based transactions or applications built on Kabira. Kabira can also replicate data in its permanent repository to any other repositories that may need to use the data on a read-only basis.
- As a high-performance cache: In this pattern, the system of record is the master and Kabira loads data from the system of record when the system is started. Then, as transactions are processed, Kabira sends back a stream of updates to the system of record.
- As a hybrid: In this model, Kabira combines aspects of both a high-performance cache and a system of record. With respect to master data, (which changes much less frequently than transactional data), Kabira may play the role of a cache and bring master data in at startup time, not updating it until the system is restarted or the data is scheduled for a refresh. Transactional data, particularly if it is being assembled from many different underlying systems, could be managed inside Kabira as a system of record. Changes to the transactional data could then be distributed to the source systems in a variety of ways previously described.

Kabira Development Methodology

Kabira’s methodology for creating solutions delivers on the full promise of model-driven development. Most other development environments or products that use model-driven development offer a partial solution, presenting a visual interface for some portions of an application, even though the objects that the model is gluing together must be developed in a traditional programming language, such as Java or C++.

While partially modeled applications can improve productivity in some situations, the approach is inadequate for high-performance computing. In a fully modeled solution - one in which all the objects and their behavior are expressed in a modeling language - a whole new world of optimization and automatic assembly of components is opened up. At compile time, Kabira has a view of the entire application and can then render an implementation of that model using its understanding of threads to take advantage of parallelism and apply all of the transactional services where needed.

The result of Kabira’s model-driven development is reduction in development and implementation complexity and an increase in application quality. The power of modeling has tremendous leverage: 20,000 lines of a model can generate the equivalent of 2 million lines of implementation code. Models are much easier to maintain than thousands of lines of code and are less prone to error.

Model-Driven Architecture Based on Eclipse

Eclipse provides a visual representation of the models used to create engines in the Kabira Transaction Platform. Kabira uses a Model-Driven Architecture (MDA) which allows applications to be specified using a modeling language like UML or BPEL and the implementation of the application can then be generated. MDA is often praised but seldom fully implemented. The Kabira Transaction Platform is the only high-performance transaction processing platform that is completely model-driven. Once the model is complete, it is then compiled into an executable engine.

Benefits of Kabira's Model-Driven Architecture: Develop Complex Solutions in Weeks, Not Months

Both UML and BEPL have Action Languages that allow the behavior of objects to be specified, so that an application can be fully modeled. In UML versions prior to 1.4, the structure of an application was described and then coded in an implementation language such as Java or C++ that created much of the objects and the data contained within them. The job of finishing creation of an application was done by adding statements to the objects that implemented the logic (the "action semantics") that did the work of the application. The introduction of Action Languages means there is no longer a need to fill in the actions of a model with another implementation language.

Kabira's model-driven architecture enables programmers to develop and deploy new services and network-based applications using high-level UML models, standard action language and automatically-generated Web Services, CORBA, Java or XML interfaces that are independent of underlying technology. This allows system architects to design very complex solutions in a few weeks or months, with a small team of modeling and domain experts. The resulting applications-which are compiled 100% from high-level models-are supported by the Transaction Platform and execute with unprecedented speed, flexibility and scalability. All applications can automatically recover from system or runtime faults without loss of transactions or data. Kabira eliminates the need for complex, 3GL coding without compromising any of the flexibility of traditional development. It enables network and transaction architects to leverage legacy applications, hardware platforms and network elements, and to focus on service delivery processes and models rather than on infrastructure requirements.

Configuration

One of the most convenient features of the Kabira Transaction Platform is the separation of configuration information from applications. The system can be configured using XML instructions, without the need for programming. The switch can also be dynamically re-configured while the system is running to allow for new business rules and multiple versions of business rules.

Kabira's configuration files can be versioned. One version, for example, could describe the testing environment, another could describe the staging environment, and still another could describe the production environment. Kabira protects against mistakes in changing configurations by refusing to accept certain changes, such as closing a connection when the other side of the connection is still open.

Simulation and Testing

Another benefit of the completely modeled applications environment is that the model can be used to generate initial test cases. The end-point simulation and testing framework allows for the creation of proxies for external services to simulate their behavior. The testing framework also contains a mechanism for storing test data. Using this framework the functional accuracy can be tested through the creation of pass/fail tests. The performance of the application can be analyzed under load. The performance testing features of the platform allow statistics to be gathered to determine how to optimize the model.

Deploying and Operating the Kabira Transaction Platform

The Kabira Transaction Platform is built to be highly configurable at run-time. Configuration changes, loading of new components and tuning of high-availability features can all take place during transaction processing without interruption. The key to successful deployment and operation of solutions based on the Kabira Transaction Platform is a clear understanding of the computing resources required for each node. Once this is understood, the highest performance architecture can be designed at the lowest possible cost.

Deployment Architecture

In a hardware-based high-availability system, every part of an application runs on expensive, high-performance hardware. If this makes sense for the most demanding applications, it can be costly overkill in situations where smaller, cheaper computers can do a fine job of handling simpler tasks with sufficient speed. Each node runs on one machine. Kabira's powerful platform node architecture can run on commodity hardware, giving you the choice of determining where to place nodes, based on the type of hardware each node requires to achieve optimum performance.

If several nodes are sharing the burden of transaction processing, the size and expense of the hardware allocated to each node can be tuned so that each node gets the CPU and memory it needs at the lowest possible cost. Kabira allows nodes that do not need such high performance hardware to run on lower-cost Linux systems.

Operational Management Tools

Kabira's tools for operational management resemble the current incarnation of UNIX command line tools. Each node can be controlled and monitored locally or remotely with set of commands. In the near future, the command-line approach will be supplemented with a graphical tool that allows multiple nodes to be managed from one unified, graphical interface.

Conclusion

Kabira is a proven technology that powers high-performance systems for some of the largest companies in the world. The mechanisms and architectural strategies explained in this white paper show how the Kabira Transaction Platform is constructed from top to bottom to eliminate the traditional trade-offs of the transaction processing triangle. Kabira's new generation technology fulfills the business imperative of the digital age: combining an extremely high performance, eminently flexible, highly scalable, ultra-fast processing environment on lowest cost platforms to create the fastest processing solutions in the world attainable at the lowest total cost of ownership.

Contact your Kabira representative to begin a conversation about how your company can improve IT performance and revenues and reduce total costs using Kabira's Transaction Platform and software solutions.

Corporate Headquarters – USA
1850 Gateway Drive, 5th Floor
San Mateo, CA 94404
Tel: 650 931 3700
Fax: 650 931 3799

Kabira Technologies – France
11, rue Scribe
75009 Paris
France
Tel: +33 1 44 51 70 90
Fax: +33 1 44 51 00 80

001-0501 – August 2006